Ecole Nationale des Sciences Appliquées

Kénitra

Cycle Préparatoire

Architecture des ordinateurs et Systèmes d'exploitation

Pr. N. EL HAMI

Année universitaire : 2020/2021

Chapitre 3 :

Systèmes de numération : Partie 1

1. Principe de numération: 3
   1.1 Objectifs: 3
   1.2 Codage d'information: 3
   1.3 Système de numération : 3
   1.4 Bases de numération (Binaire, Octale et Hexadécimale): 4
   1.5 Les bases classiques en informatiques : 5

2- Conversion et changement de base 7
   2.1 Conversion de la base décimale à la base octale: 7
   2.2 Conversion de la base décimale à la base Hexadécimale: 8
   2.3 Conversion de la base octale à la base Hexadécimale: 8
   2.4 Conversion de la base binaire à la base octale: 9
   2.5 Conversion de la base octale à la base binaire: 10
   2.6 Conversion de la base binaire à la base Hexadécimale: 10
   2.7 Conversion de la base Hexadécimale à la base binaire: 11
   2.8 Conversion de la base décimale à la base binaire: 11

3- Opérations arithmétiques 12
   3.1 Addition en binaire 12
   3.2 Soustraction en binaire 13
   3.2 Multiplication en binaire 13

4- Nombres signés 13
   4.1 Nombres négatifs (nombres signés) : 13
   4.2 Le complément à 2 CA(2) : 15
   4.3 La retenue et le débordement : 16

1. Principe de numération:

1.1 Objectifs:

Comprendre comment les ordinateurs représentent une information ?

Ils Convertissent des nombres entiers et réels en représentation binaire et *vice versa et* réalisent des opérations mathématiques de base (addition, soustraction et multiplication).

Les informations traitées par les ordinateurs sont de différentes natures : Nombres, texte, images, sons, vidéo. Ils sont représentés sous forme binaire (BIT : Binary digIT) une suite de 0 et de 1.

1.2 Codage d'information:

Le codage de l'information permet d'établir une correspondance qui permet sans ambiguïté de passer d'une représentation (dite externe) d'une information à une autre représentation (dite interne : sous forme binaire), suivant un ensemble de règles précises.

Exemple : • Le nombre 65 : 65 est la représentation externe du nombre soixante cinq • La représentation interne de 65 sera une suite de 0 et 1 (1000001)

En informatique, Le codage de l'information s'effectue principalement en trois étapes :

- L'information sera exprimée par une suite de nombres (Numérisation)
- Chaque nombre est un code sous forme binaire (suite de 0 et 1)
- Chaque élément binaire est représenté par un état physique

1.3 Système de numération :

Le système de numération décrit la façon avec laquelle les nombres sont représentés. Un système de numération est défini par :

- Un alphabet : ensemble de symboles ou chiffres, Des règles d'écritures des nombres en réalisant une Juxtaposition de symboles

Exemple : Numération décimale

Le système de numération décimale est composé de dix chiffres A= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

Le nombre 10 est la base de cette numération. B = 10.

C'est un système positionnel : chaque position possède un poids.

Par exemple, le nombre 8251 s'écrit comme :

$$8251 = 8*10^3+2*10^2+5*10^1+1*10^0$$

1.4 Bases de numération (Binaire, Octale et Hexadécimale):

Système binaire utilise deux chiffres : {0,1}

- ⇒ C'est avec ce système que fonctionnent les ordinateurs

Système Octale utilise huit chiffres : {0,1,2,3,4,5,6,7}

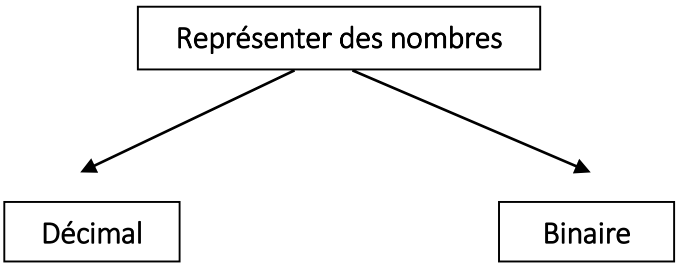
- ⇒ Elle permet de coder 3 bits par un seul symbole.

Système Hexadécimale utilise 16 chiffres : {0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F}

- ⇒ Elle permet de coder 4 bits par un seul symbole

Résumé :

Les ordinateurs calculent en binaire. Il s'agit de la base 2 dans laquelle seuls les symboles 0 ou 1 ; éteint ou allume, ouvert ou ferme ; noir ou blanc ; haut ou bas.



Généralisé vers une Base quelconque B

Symbole : Décimal : On utilise dix chiffres (0,1,2,.......9)

Binaire (0,1) deux chiffres

Il y a B Symbole différents pour exprimer un nombre en une base B

- Si **B < 10** on utilise les chiffres (0,1,2,…….9) exemple B=8 on utilise les chiffres 0-7
- Si **B > 10** in faut définir des chiffres entre 10 et B=16 on utilise les let

a, b, c, d, e, f
10,11,12,13,14,1

Pour représenter un nombre X de n « chiffres »

« $X_{n-1}\ X_{n-2}\ X_{n-3}$ ……..$X_1\ X_0$ »

X = $X_{n-1}\ B^{n-1} + X_{n-2}\ B^{n-2} +$ ……………+ $X_1\ B + X_0$

X = $\sum_{i=0}^{n-1} X_i * B^i$

Quelle est la valeur maximale dans une base B à n chiffres ?

Réponse = $\boldsymbol{B^n} - 1$

Démonstration :

Pour n symboles → tous les symboles sont égales à (B-1) (9 dans la base 10)

(B-1) B^{n-1} + (B-1) B^{n-2} + ……..+ (B-1) B + (B-1) = (B-1) $\sum_{i=0}^{n-1} B_i = \frac{B^n - 1}{B-1}$. (B-1)

= $B^n - 1$

1.5 Les bases classiques en informatiques :

Base B=2 calcul binaire Première base fondamentale

Un nombre binaire peut prendre 0 ou 1 il s'appelle un BIT (BInary digiT)

Exemple n=4 Nombre B=2 1 1 0 1 en Décimal?

Position	3 2 1 0
Valeur	$2^3\ 2^2\ 2^1\ 2^0$
Nombre	1 1 0 1

Calcul	1 x 2^0 = 1
	0 x 2^1 = 0
	1 x 2^2 = 4
	1 x 2^3 = 8

Total	13

Base B=8 octale

Un nombre octal peut prendre de 0 1 2 3 4 5 6 7

Exemple n=5 Nombre B=8 3 5 2 4 6 en Décimal?

Position	4 3 2 1 0
Valeur	8^4 8^3 8^2 8^1 8^0
Nombre	3 5 2 4 6

Calcul

$6 \times 8^0 = 6$

$4 \times 8^1 = 32$

$2 \times 8^2 = 128$

$5 \times 8^3 = 2560$

$3 \times 8^4 = 12288$

Total 15014

Base B= 16 hexadécimale

Un nombre hexadécimal peut prendre 0 1 2 3 4 5 6 7 8 9 A B C D E F

Un nombre hexadécimal peut prendre	0 1 2 3 4 5 6 7 8 9	A	B	C	D	E	F
Position	0 1 2 3 4 5 6 7 8 9	10	11	12	13	14	15

Exemple n=3 Nombre B=16 A 3 2 en Décimal?

Position	2 1 0
Valeur	16^2 16^1 16^0
Nombre	A 3 2

Calcul

$2 \times 16^0 = 2$

$3 \times 16^1 = 48$

$10 \times 16^2 = 2560$

Total 2610

2- Conversion et changement de base

2.1 Conversion de la base décimale à la base octale:

$625_{10} = ?_8$

Position	...	3	2	1	0
Valeur	...	8^3	8^2	8^1	8^0
Ou	...	512	64	8	1

625	8
1	78

78	8
72	9
6	

9	8
1	1

$625_{10} = 1161_8$

$625_{10} = 1 \times 8^3 + 1 \times 8^2 + 6 \times 8^1 + 1 \times 8^0$

$625_{10} = 1 \times 512 + 1 \times 64 + 6 \times 8 + 1$

15014	8
15008	1876
6	

1876	8
1872	234
4	

234	8
2	29

29	8
24	3
5	

$15014_{10} = 3\ 5\ 2\ 4\ 6$

Calcul

$6 \times 8^0 = 6$

$4 \times 8^1 = 32$

$2 \times 8^2 = 128$

$5 \times 8^3 = 2560$

$3 \times 8^4 = 12288$

Total 15014

2.2 Conversion de la base décimale à la base Hexadécimale:

$421_{10} = ?_{16}$

Position	...	3	2	1	0
Valeur	...	16^3	16^2	16^1	16^0
Ou	...	4096	256	16	1

$$\begin{array}{r|l} 421 & 16 \\ \cline{2-2} 416 & 26 \\ \cline{1-1} 5 & \end{array} \qquad \begin{array}{r|l} 26 & 16 \\ \cline{2-2} 16 & 1 \\ \cline{1-1} 10 & \end{array}$$

$421_{10} = 1A5_{16}$

421_{10} = 1 x 16^2 +10 x 16^1 + 5

421_{10} = 1 x 256 + 10 x 16 + 5

$$\begin{array}{r|l} 2610 & 16 \\ \cline{2-2} 2608 & 163 \\ 2 & \end{array} \qquad \begin{array}{r|l} 163 & 16 \\ 160 & 10 \\ \cline{2-2} 3 & \end{array}$$

$2610_{10} = A32_{16}$

- -

Calcul 2 x 16^0 = 2

3 x 16^1 = 48

10 x 16 = 2560

- - - - - - - - - - - - -

Total 2610

2.3 Conversion de la base octale à la base Hexadécimale:

A- 1ére Méthode

$217_8 = ?_{16}$

217_8 = 2 x 8^2 + 1 x 8^1 + 7 x 8^0

= 128 + 8 + 7 = 143

$= \mathbf{143_{10}}$

143	16
128	8
15	

$= \mathbf{8F_{16}}$

$\mathbf{217_8 = 143_{10} = 8F_{16}}$

B- 2ème Méthode

$\mathbf{217_8} = ?_{16}$

$\mathbf{217_8} = 2 \times \mathbf{8^2} + 1 \times \mathbf{8^1} + 7 \times \mathbf{8^0}$

$= 2 \times 8 \times 8 + 8 + 7$

$= 8 \times 16 + 15$

$= 8 \times \mathbf{16^1} + 15 \times \mathbf{16^0}$

$\mathbf{217_8} = \mathbf{8F_{16}}$

2.4 Conversion de la base binaire à la base octale:

$\mathbf{100010101_2} = ?_8$

A- 1ére Méthode

Position 8 7 6 5 4 3 2 1 0

1 0 0 0 1 0 1 0 $\mathbf{1_2}$

$= 1 \times \mathbf{2^8} + 1 \times \mathbf{2^4} + 1 \times \mathbf{2^2} + 1 \times \mathbf{2^0}$

$= (\mathbf{2^3})^2 \times \mathbf{2^2} + \mathbf{2^3} \times 2 + 1 \times \mathbf{2^2} + 1 \times \mathbf{2^0}$

$= (\mathbf{2^3})^2 \times 4 + \mathbf{2^3} \times 2 + 5$

$\mathbf{100010101_2} = \mathbf{425_8}$

B- 2ème Méthode

2 1 0	2 1 0	2 1 0
1 0 0	0 1 0	1 0 1_2
1 x 2^2	1 x 2^1	1 x 2^2 + 1 x 2^0
4	2	5

$100010101_2 = 425_8$

2.5 Conversion de la base octale à la base binaire:

$425_8 = ?_2$

4	2	5
1 x 2^2	1 x 2^1	1 x 2^2 + 1 x 2^0
1 0 0	1 0	1 0 1_2
1 0 0	0 1 0	1 0 1_2

$425_8 = 100010101_2$

2.6 Conversion de la base binaire à la base Hexadécimale:

$10001010_2 = ?_{16}$

A- 1ére Méthode

Position 7 6 5 4 3 2 1 0

1 0 0 0 1 0 1 0

= 1 x 2^7 + 1 x 2^3 + 1 x 2^1

= 2^4 x 2^3 + 1x 2^3 + 1 x 2^1

= 2^4 x 8 + 8 + 2

= 16^1 x 8 + 16^0 x 10

$$\mathbf{10001010_2} = \mathbf{8A_{16}}$$

B- 2ème Méthode

3 2 1 0	3 2 1 0
1 0 0 0	1 0 1 0
1 x 2^3	1 x 2^3 + 1 x 2^1
8	10

$$\mathbf{10001010_2} = \mathbf{8A_{16}}$$

2.7 Conversion de la base Hexadécimale à la base binaire:

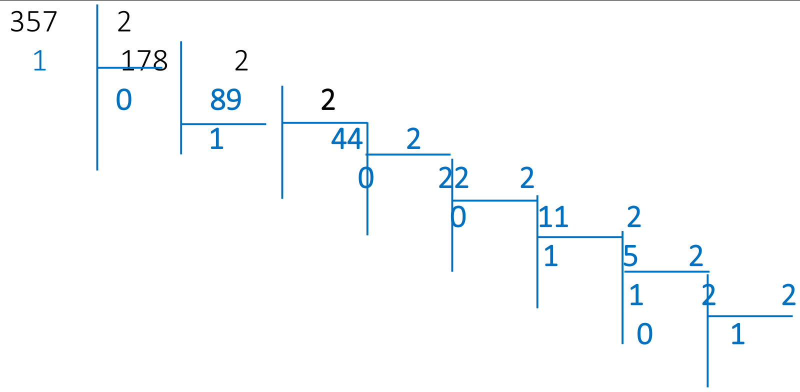
Exemple : $\mathbf{23D5_{16}}$

2	3	D	5
3 2 1 0	3 2 1 0	3 2 1 0	3 2 1 0
1 0	1 1	1 1 0 1	1 0 1
0010	0011	1101	0101

$$\mathbf{10001111010101_2}$$

2.8 Conversion de la base décimale à la base binaire:

$\mathbf{357_{10}}$ = $?_2$



$\mathbf{357_{10}} = \mathbf{101100101_2}$

3- Opérations arithmétiques

3.1 Addition en binaire

0 + 0 = 0 retenue 0

0 + 1 = 1 retenue 0

1 + 0 = 1 retenue 0

1 + 1 = 0 retenue 1

Retenue 11 1 1

$\mathbf{101011_2}$ $= \mathbf{43_{10}}$

+

$\mathbf{1101110_2}$ $= \mathbf{110_{10}}$

= $\mathbf{10011001_2}$ $= \mathbf{153_{10}}$

3.2 Soustraction en binaire

0 - 0 = 0 retenue 0

0 - 1 = 1 retenue 1

1 - 0 = 1 retenue 0

1 - 1 = 0 retenue 0

$$
\begin{array}{lll}
 & \mathbf{1101110_2} & = \mathbf{110_{10}} \\
 & \;\;0\,0\,1\,1 & \\
- & & \\
 & \mathbf{101011_2} & = \mathbf{43_{10}} \\
\hline
= & \mathbf{1000011_2} & = \mathbf{67_{10}}
\end{array}
$$

3.2 Multiplication en binaire

$$
\begin{array}{lll}
 & \mathbf{10110_2} & = \mathbf{22_{10}} \\
\times & & \\
 & \mathbf{1011_2} & = \mathbf{11_{10}} \\
\hline
+ & 10110 & \\
+ & 10110- & \\
+ & 00000-- & \\
 & 10110--- & \\
\hline
= & 11110010 &
\end{array}
$$

4- Nombres signés

4.1 Nombres négatifs (nombres signés) :

Nous avons présenté les nombres entiers positifs

Comment représenter les nombres négatifs?

Le signe « - » ne peut être représenté sur une case un BIT (0,1).

Représentation:

Le bit de poids le plus fort et appelé aussi bit de signe one le représente par 1.

Exemple :

Sur 8 BIT $\mathbf{47_{10} = 101111_2}$

1	0	1	0	1	1	1	1

Bit de

Intervalle de représentation:

Sur n BIT 2^n possibilité [0, $\mathbf{2^n}$- 1]

Pour représenter les nombres négatifs, on a défini un intervalle [-$\mathbf{2^{n-1}}$, $\mathbf{2^{n-1}}$- 1]

Exemple :

n=8 sur 2^8 = 256 *combinaisons* [0, 255]

Pour représenter les nombres négatifs, on a défini un intervalle [$\mathbf{-128, 127}$]

L'arithmétique binaire :

L'opération d'addition avec la notation d'un nombre négatif sera erronée (voir l'exemple ci-dessous), d'où la nécessité d'utilisé une méthode pour remédier à ce problème. La solution utilisée est le complément à 2.

Exemple :

n=8

$3_{10} - 3_{10} = 3_{10} + (-3)_{10}$

= 0000 0011 + 1000 0011

```
  0000 0011
  1000 0011
-----------------
  1000 0110
```

1000 0110 = $\mathbf{(-6)_{10}}$

$3_{10} - \cancel{3_{10} = (-6)_{10}}$ Fausse

4.2 Le complément à 2 CA(2) :

Le complément à 2 est une représentation standard sur les ordinateurs pour exprimer les nombres entiers négatifs.

→ La représentation signée ou d'entiers signés sont toujours exprimer en complément à 2

<u>Les étapes à suivre pour obtenir le complément à 2:</u>

$(-52)_{10}$

1	0	1	1	0	1	0	0

CA(1) Complément à 1 = inversion des bits

1	1	0	0	1	0	1	1

CA(2) Complément à 2 = CA(1) + 1 + $\boxed{1}$

1	1	0	0	1	1	0	0

Exemple :

n=8

3_{10} - 3_{10} = 3_{10} + $(-3)_{10}$ = 3_{10} + CA2 (-3)

$3_{10} =$ 0000 0011

CA1 11111100

\+ 1

CA2(-3) = 11111101

3_{10} 0 0 0 0 0 0 1 1

CA2(-3) + 1 1 1 1 1 1 0 1

1

----

0 0 0 0 0 0 0 0 = $(\mathbf{0})_{10}$

La retenue

3_{10} + $(-3)_{10}$ = $(\mathbf{0})_{10}$

On a repris l'exemple précèdent avec le complément à 2, le résultat obtenu est vrai $(\mathbf{0})_{\mathbf{10}}$. De cette manière on a résolu l'opération arithmétique souhaitée.

4.3 La retenue et le débordement :

- On appelle une retenue lorsqu'une opération arithmétique génère un report.
- On appelle un débordement ou dépassement de capacité lorsque le résultat de l'opération arithmétique sur n bits et faux (dépasse n bits). Le nombre de bits utilisés qui est insuffisant pour contenir le résultat.